# automotiveHMI – Model-Based In-Vehicle Infotainment Description as a Stepping Stone for the Integration of Car and Web

Gerrit Meixner
German Research Center for Artificial Intelligence (DFKI)
Trippstadter Strasse 122
67663 Kaiserslautern
Gerrit.Meixner@dfki.de
+49 631 205 75-3415

Marius Orfgen
German Research Center for Artificial Intelligence (DFKI)
Trippstadter Strasse 122
67663 Kaiserslautern
Marius.Orfgen@dfki.de
+49 631 205 75-3413

Moritz Kuemmerling
German Research Center for Artificial Intelligence (DFKI)
Trippstadter Strasse 122
67663 Kaiserslautern
Moritz.Kuemmerling@dfki.de
+49 631 205 75-3414

## Statement of interest

Different studies show (e.g., [1]) that over 80% of today's innovations in the automotive industry are based on car electronics and its software. These innovations can be categorized into hidden technologies (e.g., ASP, ESP), comfort functions (e.g., navigation, communication, entertainment) or driver assistance (e.g., distance checking). Especially the last two categories have to be configurable by the driver and therefore require a certain amount of driver interaction. This results in a need for a modern and consistent human-machine-interface (HMI) which on one hand allows the configuration of these systems but which on the other hand conforms to the specialized requirements of the automotive industry. Some of these requirements are:

- The interaction devices have to be integrated into a limited space.
- The HMI has to be intuitively usable and adaptable, since drivers generally do not get an extensive explanation.
- The HMI has to be very easy to use and should distract the driver as little as possible from his main task of driving.

Additionally to the growing number of configurable systems in a modern car, there is a need for shorter release cycles and lower costs for the development of HMIs. One main source of problems in the development of HMIs is the communication overhead caused by informal specifications [2]. This overhead is needed to reduce the ambiguity and the incorrectness of the specification document. A more formal approach to specification might reduce this overhead dramatically, leading to shorter development times, cost savings and fewer problems.

The publicly funded research project automotiveHMI (http://www.automotive-hmi.org) consists of 11 partners of the German automotive industry, two research institutes as well as one association. The industry members are car manufacturers (Original Equipment Manufacturer - OEM), suppliers and tool developers which together cover the complete development chain for automotive HMI-systems [3].

The research project is based on three pillars. The main pillar is represented by the development of a model-based interchange format for the specification of automotive HMI-systems. Current HMI development processes are characterized by different, inconsistent workflows and heterogeneous tool chains. The exchanged requirements are often inconsistent, redundant, incomplete and in general not machine-readable.
These circumstances lead to tremendous communication efforts between OEMs and their suppliers until both get the same understanding of the new HMI-system. Mistakes and bugs are often first noticed when the supplier delivers the first software-version back to the OEM. At that late time modifications are very cost intensive and change-request negotiations become a common annoyance [4].

A further problem is the wide range of actors from many different branches that are involved in the automotive HMI development: Computer scientists and electrical engineers work together with designers, ergonomists and psychologists in interdisciplinary teams. The HMI modeling language that is to be developed shall serve as the connective link (lingua franca) between these actors.
On this account the HMI modeling language has to be domain specific. Domain specific languages (DSL) are dedicated to a particular problem domain and their "vocabulary" is generally based upon common expressions that are typical for the domain. Thus DSLs are far more expressive in their domain than general-purpose languages would be.

The project goal of automotiveHMI is to create a modeling language that will allow the different members of the development chain to specify and exchange the requirements and artifacts for HMI development. The language should be established as an industry standard to improve communication and the development of interoperable tools.
This modeling language forms a common data interface for the different process members (e.g. OEMs, suppliers). It allows bridging the "digital gap" currently found in today's HMI development due to the document-driven exchange of specification data. Building on this data interface, it is possible to exchange digital information for use in development tools without information breaks. This results in a more efficient collaboration of development members and allows further the digital convergence of different HMI subsystems (e.g. navigation, communication, car systems).

The project automotiveHMI uses concepts from the field of model-based user interface development (MBUID) (http://www.w3.org/2011/mbui/) to create a modeling language that meets the requirements of formality and correctness. For further information concerning MBUID see [5], [6] and [7].

The introduction of a model-based interchange format will serve as an interface between the process participants thus avoiding media discontinuity and improving the communication among the involved actors. The interchange format (usually a XML-dialect) allows describing layout, structure and behavior of the HMI-system independently of the final target-code and target-system.

## Concrete Examples of our suggestions

In order to allow future cars to harness the benefits of internet connections and access to remote services, it is important to transform today's in-vehicle infotainment (IVI) systems by opening them up for service integration as well as an "app concept" similar to that found in smartphones and tablets. This transformation marks a turning point in the development of IVI solutions, since they will no longer be closed systems with a predefined set of functions but more of a standardized middleware that allows third parties to implement new functions and services.

Today's infotainment solutions are developed as a closed system. At the beginning of the development process, the functionality is defined and stays constant over the process and the time after SOP (start of production). Software updates after SOP consist of bug fixes and other data (e.g. more recent map data), with the explicit goal to keep the user interface the same (so that the consumer does not notice any changes after the update). Two to three years after start of development, the infotainment system is put into new cars. This means that a newly released car contains an infotainment system which functionalities are based on projections that were made years ago. In today's fast-moving world, where smartphone and tablet apps are updated on a monthly basis (or faster), these systems look and feel outdated to the modern technology-savvy user.

The development processes and tools that are currently used in IVI development need to be adapted for the rapid development or improvement requirements of the future. This means adding, changing or removing functionality, switching to different hardware platforms and supporting external platforms (e.g. tablets).

The automotiveHMI project represents the first step into this direction. The interchange format developed in the project aims to provide a technology-independent, high-level description format for IVI systems, regardless of the hardware, operating system or middleware of the intended target(s).

This interchange format bridges the gap between completely formal specifications and informal, human-readable descriptions of a car infotainment system. In contrast to older approaches, which forced the designers so specify everything completely formal (even things where they had no knowledge of, e.g. middleware communication), the idea is to gather all information about an infotainment system in one model, regardless of the formality level, and then have experts for different topics (e.g. hardware drivers, animations, color spaces) refine the parts that are still informal until the model has reached a sufficient level of formality. This level depends on the requirements of the model, a model from which code is to be generated has higher requirements than a model where only the pixel positions of the graphical elements should be exact.

Similar to HTML, which serves as a description independent of platform or operating system, the interchange format can be read and written without deeper knowledge of the middleware and hardware specifics (except for obvious cases like screen resolution or color depth) and is then either interpreted or compiled to create a working user interface. For systems with lower processing power, binary code might be generated, while systems with more power (e.g. tablets) could process the format a runtime. Similar to HTML in conjunction with CSS, the interchange format describes the look and feel of the displayed contents, but also allows describing middleware interaction in a middleware-independent manner. A code generator for Java might therefore create code to access a messaging middleware component, while a generator for HTML5 would create the necessary WebSocket components on client and server side.

A great benefit of the interchange format is that different roles (designers, developers) and companies (OEMs, suppliers, third-party companies) can work together to create infotainment systems at a much higher speed. Moreover, by using code generation, even greater changes to the system can be introduced without having to edit a large code base.

Introduction of the interchange format into automotive development processes will lower the barrier for integrating web services or external user interfaces (e.g. on tablets or smartphones) and will allow the development of automotive web apps.

The explicit modeling of the infotainment system allows the generation of user interfaces for different platforms and technologies while maintaining a machine-readable, target-independent description of the final user interface and related functionality. Since the static and dynamic aspects of the HMI are described in a model that can be analyzed for consistency and completeness, it can be checked much easier compared to source code.

A next step after the integration of the interchange format into the development processes would be to decouple the development process of the car from that of the infotainment system. Infotainment concepts and trends change at a much higher speed than those of traditional car development. By providing open infotainment hardware, new functionality and new services could be created and deployed to cars at a speed similar to that of smartphones and tablets.

Users of modern consumer devices are accustomed to (and expect) updates of their systems and applications. In conjunction with the web-enabled, connected car, the opportunity to update applications in a similar frequency as on smartphones and tablets will lead to greater customer satisfaction and may also add new revenue streams for the OEMs and third-party companies, since car infotainment apps can be sold and distributed similar to smartphone/tablet apps. Moreover, the integration of web-based services like speech-to-text translation, cloud storage, live media streaming or advanced route guidance allows selling services that will generate revenue over the lifetime of the car.

To demonstrate these ideas, work is currently underway to implement a tool chain that allows rapid specification of infotainment systems with prototyping tools and image manipulation programs. This information is then imported into the interchange format and enriched with event information describing the interaction between the user interface parts and the middleware.

The model created by the importer can then be used to generate the code for different targets. The tool chain currently supports the generation of Java code as well as the creation of HTML5 user interface. The HTML5 interface looks and behaves exactly like the Java one, both adhering to the description in the model. The middleware communication is achieved by different means depending on the target, with the Java code using simple method calls while the HTML5 UI uses WebSockets to perform the bidirectional communication. Therefore a tablet with a modern browser can actually replace or extend the infotainment UI integrated into the car.

Work is currently underway to write an exporter to Elektrobit GUIDE, a tool used in the automotive industry to create code for the target hardware in the car. Once this exporter is finished, it will be possible to create a UI using designer tools and automatically export it to smartphones, tablets and the car target hardware. This direct export allows designers to experience their ideas directly, leading to much faster development cycles compared to the current practice of waiting weeks or months until the supplier created a user interface that matches the designers' descriptions.

## Acknowledgements

The work described in this position paper is conducted within the project automotiveHMI. automotiveHMI is funded by the German Federal Ministry of Economics and Technology under grant number 01MS11007.

## 1. REFERENCES

[1] Danneberg, J. and Burgard, J. (2007) A comprehensive study on innovation in the automotive industry. http://www.oliverwyman.com/pdf_files/CarInnovation2015_engl.pdf (Retrieved October 10, 2012).

[2] Huebner, M. and Grüll, I. (2007) ICUC-XML Format. Format Specification Revision 14. Elektrobit.

[3] Kuemmerling, M. and Meixner, G. (2010) Model-Based User Interface Development in the Automotive Industry. Proc. of the 3rd International Workshop on Multimodal Interfaces for Automotive Applications (MIAA), 41-44, Palo Alto, USA.

[4] Kuemmerling, M. and Meixner, G. (2011) automotiveHMI – a model-based interchange format for optimized HMI-development in the automotive industry, ATZ elektronik, August 2011, 52-55, Springer.

[5] Cantera Fonseca, J. M., González Calleros, J. M., Meixner, G., Paternó, F., Pullmann, J., Raggett, D., Schwabe, D., and Vanderdonckt, J. (2010) Model-Based UI XG Final Report, W3C Incubator Group Report.

[6] Hussmann, H., Meixner, G., and Zuehlke, D. (2011) Model-Driven Development of Advanced User Interfaces, Studies in Computational Intelligence, Vol. 340, Springer, Heidelberg.

[7] Meixner, G., Paternó, F., and Vanderdonckt, J. (2011) Past, Present, and Future of Model-Based User Interface Development. i-com, 10, 3, 2-11.